Feeling Validated: Constructing Validation Sets for Few-Shot Intent Classification

Anonymous EMNLP submission

Abstract

We study validation set construction via data augmentation in true few-shot intent classification. Empirically, we demonstrate that with scarce data, model selection via a moderate number of generated examples consistently leads to higher test set accuracy than either model selection via a small number of held out training examples, or selection of the model with the lowest training loss. For each of these methods of model selection—including validation sets built from task-agnostic data augmentation—validation accuracy provides a significant overestimate of test set accuracy. To support better estimates and effective model selection, we propose PANGEA, a generative method for domain-specific augmentation that is trained once on out-of-domain data, and then employed for augmentation for any domain-specific dataset. In experiments with 6 datasets that have been subsampled to both 5 and 10 examples per class, we show that PANGEA is better than or competitive with other methods in terms of model selection while also facilitating higher fidelity estimates of test set accuracy.

1 Introduction

Model selection is a key step in machine learning (ML) workflows. In typical model development, training is initiated with many hyperparameter configurations, which results in many distinct models. The performance of a model is highly sensitive to these hyperparameters (Dodge et al., 2020). For example, when prompting large language models, some orderings of a given set of samples leads to state-of-the-art results while other orderings of the same samples lead to results that resemble random guessing (Lu et al., 2022).

In few-shot learning, i.e., learning with only a handful of training examples, effective model selection is more critical and challenging than in settings with larger data sets. In modern ML, model selection is typically performed by evaluating each model on a validation set and choosing the model that performs best, according to some metric of interest. Model selection in true few-shot settings is challenging because in these settings there is no validation set (Perez et al., 2021; Bragg et al., 2021). While it is possible to hold out a portion of training data for use as a validation set, in few-shot settings this is problematic for two reasons. First, holding out data when examples are scarce can dramatically worsen training. Second, since the number of held out examples is necessarily small, the examples chosen for validation constitute a high variance estimator of model performance, and thus can lead to poor model selection. While there are methods of model selection that do not require validation sets, such as cross-validation and minimum description length (Rissanen, 1978), recent work demonstrates that neither are dependable selectors of high-performing deep models in few-shot settings (Perez et al., 2021). Were a validation set available, previous work shows that it can be used to consistently select high-performing models.

Given the importance of model selection in few-shot learning, and the benefit of having a moderately sized validation set, we propose to construct validation sets via data augmentation. We first study Easy Data Augmentation (EDA), a simple method of data augmentation that generates new instances by perturbing existing examples (Wei and Zou, 2019). Since examples generated by EDA are similar to the training examples, they are likely to provide good estimates of model performance on in-distribution data. On the other hand, they are likely to provide poor estimates on out-of-distribution data. Moreover, by virtue of their similarity to the training data, optimizing for examples generated by EDA could lead to overfitting.

To address these concerns, we design PANGEA, the **P**rompt and **G**uide word **A**ugmentation algorithm for training generative models for true few-shot classification settings. Critically, PANGEA trains a text generator with domain-agnostic, publicly available data; and none of the provided domain-specific data. This is important because it means that the generator is independent of the number of provided training examples–which we assume is small. After the generator is trained, it is prompted with available domain-specific data in order to generate in-domain examples. The generator also takes a set of guide words as input, which provide further control over its generations. As a result, models trained by PANGEA can create a more diverse set of examples than methods based on perturbation like EDA, thus reducing the chance of overfitting. PANGEA does not rely on filtering or feature-space interpolation, which are critical components of previously proposed methods, but unrealistic in true few-shot learning because they require model training and selection before creating new examples (Anaby-Tavor et al., 2020; Zhou et al., 2022; Kumar et al., 2019).

We experiment with 4 styles of model selection and 6 intent classification data sets. We study intent classification because it is a prevalent problem that typically manifests in the true few-shot setting (Coucke et al., 2018; Kumar et al., 2019). Our experiments reveal that model selection with synthetic data (built by EDA or PANGEA) yield better models than selection with held out data or the training loss. Interestingly, while EDA was shown to provide negligible performance gains when used for training set augmentation (Longpre et al., 2020), our results show that it is effective when used to create validation sets for model selection. We also show that for validation sets built by PANGEA, validation accuracy of the selected model is the most reliable estimator of test set accuracy. For the other methods, the selected model's validation accuracy overestimates test set accuracy because those validation examples resemble the training data too closely. Finally, our experiments reveal that for PANGEA, the reliability of validation set accuracy is preserved across all models (i.e., all hyperparameter configurations)—not only the selected model.

2 PANGEA

Training a state-of-the-art model typically requires a large amount of data. When data is scarce, one popular approach is to generate additional data via augmentation. Task-agnostic augmentation, like EDA (Wei and Zou, 2019), can be leveraged, but these methods tend to generate examples with limited diversity. As such, these methods are ineffective when used for training set augmentation for state-of-the-art transformer models (Longpre et al., 2020). Task-specific techniques have also been proposed, however the efficacy of these methods depends on the small amount of available data. Moreover, proposed techniques rely on filtering and/or feature-space interpolation, both of which imply that training and model selection have already been performed (Anaby-Tavor et al., 2020; Zhou et al., 2022; Kumar et al., 2019). Since we are concerned with settings in which no validation data is available, these methods are inappropriate.

In this section, we describe PANGEA, an algorithm for training a generative model for text. Generators trained with PANGEA are intended for use in few-shot, domain-specific settings. Since we assume a very limited amount of domain-specific data, PANGEA trains a text generator on unlabeled, out-of-domain data. After training, any available domain-specific data is used to prompt the model to generate in-domain examples. We begin with an overview of the generator. Then we discuss PANGEA training and finally, how to use the trained generator for domain-specific example creation.

2.1 PANGEA-trained Generators

At a high-level, a PANGEA-trained generator is a model that takes two strings as input and generates a string as output. The first input, p, which we call the *prompt*, is a clause that embodies the style and content that the model's output should exhibit. The second input is a variable length sequence of *guide words*, $\mathbf{w}$ (Pascual et al., 2021). Guide words are tokens that the model is trained to include in the output, thus providing additional control over the generation. While the guide words appear in the input and output of all of the generator's training examples, the model is not forced to include all guide words in its generations.

2.2 Training

Consider a few-shot, k-way, text classification data set $\mathcal{X} = \{(x_i, y_i)\}_{i=0}^{N}$, where $y \in \{c_0, c_1, \ldots, c_k\}$ and let g be a PANGEA-trained generator, $g : p \times \mathbf{w} \rightarrow z$. In the PANGEA algorithm, the generator, g, is trained from a set of triples $\mathcal{Q} = \{(p_i, \mathbf{w_i}, z_i)\}_{i=1}^{M}$, where g must generate z_i, called the *target*, from inputs p_i and $\mathbf{w_i}$. The generator's training data, $\mathcal{Q}$, does not include any utterances from $\mathcal{X}$. Instead, examples in $\mathcal{Q}$ are

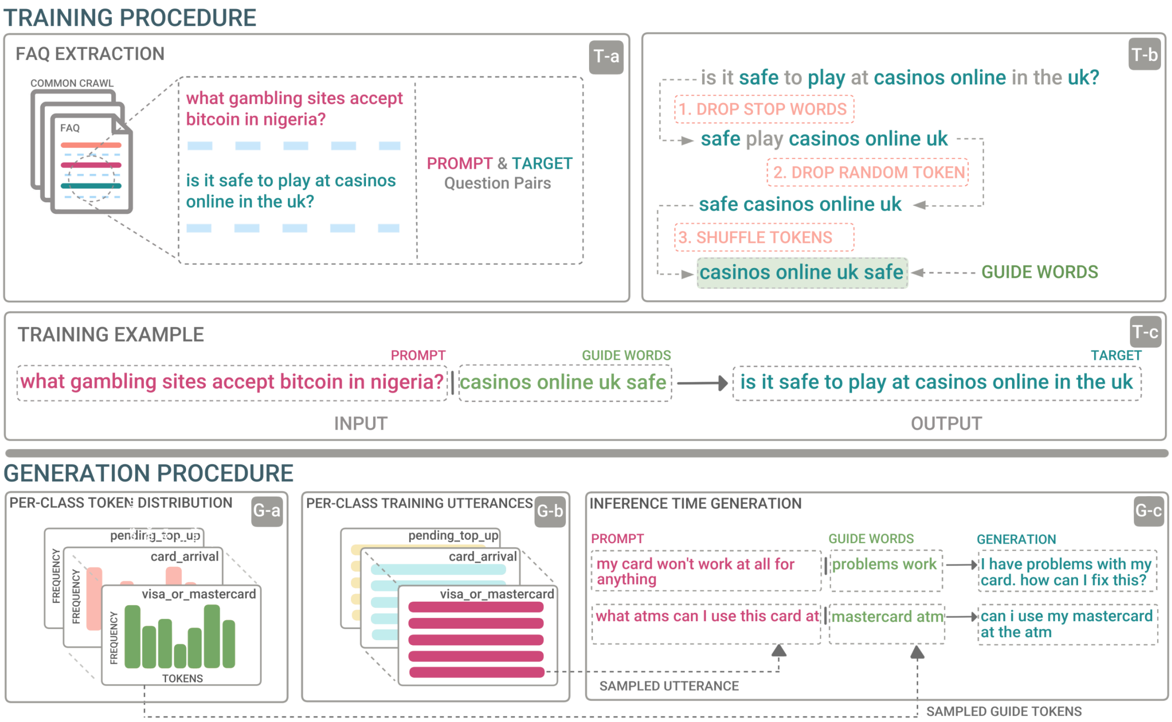


Figure 1: **Training and Generation with PANGEA.** *Prompt* and *Target* question pairs are extracted from Common Crawl (T-a). For each pair, a set of guide words is sampled from the the target (T-b). Training examples are constructed by concatenating the prompt and guide words and mapping them to the corresponding target (T-c). To generate new data, first, per-class token distributions are constructed from the few-shot data (G-a). Then, an utterance from class c is sampled uniformly (G-b). Finally, guide words are sampled from the token distribution for class c. The sampled utterance and guide words are concatenated and input to the trained generator, which produces a new training example (G-c).

constructed from a public data source, such as Wikipedia or Common Crawl. By virtue of its domain-agnostic training data, a PANGEA-trained generator is trained once and then employed for any number of tasks.

For a training example, $(p, \mathbf{w}, z)$, the prompt, p, and target, z, should be stylistically and semantically related. That way, when given a prompt in a specific domain, the generator learns to produce an output in the same domain. Moreover, the guide words, $\mathbf{w}$, should appear in z.

Formally, let $\mathcal{J}$ be a collection of utterances (e.g., sentences in Wikipedia) and let $s : \mathcal{J} \times \mathcal{J} \rightarrow \{0, 1\}$ be a binary function that returns 1 if its inputs are similar. An example of s is a function that returns 1 when two utterances appear on the same webpage. To construct an example $(p, \mathbf{w}, z)$, we select two similar utterances (with respect to s). The first we set to be p; the second, z. The guide words, $\mathbf{w}$, are (a subset of) the non-stopwords in z.

In our work, examples in $\mathcal{Q}$ are constructed from questions that appear in Common Crawl. We set s to be the function that returns 1 if two questions appear on the same web page (e.g., in the same FAQ). To construct training examples, we randomly select two questions from the same webpage to serve as the prompt, p, and target, z, respectively (Figure 1, T-a). We only utilize questions (and not answers) because the questions share some stylistic characteristics with typical utterances in intent classification. The guide words, w, are a randomly selected 95% of the non-stop word tokens in z[1] (Figure 1, T-b). We use 95% of the non-stopwords (instead of all non-stopwords) so that the model does not learn that the guide words represent all non-stopwords in the desired output. In our work, g is parameterized by T5 (Raffel et al., 2020), a large-scale, sequence to sequence model. As such, the inputs p_i and $\mathbf{w_i}$ are concatenated, but delimited by a pipe (" | ") (Figure 1, T-c). We fine-tune T5 on the constructed sequence-to-sequence examples for 20k steps with a batch size of 16. Details on question extraction from Common Crawl are

[1] w is ordered arbitrarily.

included in Appendix A.

2.3 Generation

To generate a new example of a class, c, we must choose a prompt, p, and guide words, $\mathbf{w}$. Let $X[c] = \{x_j : (x_j, y_j) \in \mathcal{X}, y_j = c\}$ be the subset of utterances in $\mathcal{X}$ of class c. In practice, we choose a prompt uniformly at random among utterances of class c, i.e., $p \sim \mathcal{U}(X[c])$ (Figure 1 G-b). Next, we select guide words. To do so, we begin by building a per-class token distribution. That is, for each utterance in $X[c]$, we filter all stop words with `spaCy` (Honnibal et al., 2020), and compute the empirical distribution of the remaining tokens (Figure 1 G-a). To sample guide words for a class c, we first sample a length L from the empirical distribution of the lengths of utterances in $X[c]$, and then sample L guide words independently from the per-class token distribution for c. The sampled prompt and guide words are concatenated (but delimited by a pipe) to form an input to the generator (Figure 1 G-c).

3 Experiments

Recall that our goal is to devise an effective method of model selection for true few-shot intent classification. To this end, we study various approaches for constructing validation sets. In this section, we present an empirical study of model selection using the constructed validation sets. We report and analyze test set accuracy achieved by selected models. We also measure the error incurred by employing validation accuracy as an estimate of test accuracy.

3.1 Setup

Datasets: Experiments are performed with the following datasets: **clinc**, **bank**, **snips**, **curekart**, **powerplay**, and **mattress** (Larson et al., 2019; Casanueva et al., 2020; Coucke et al., 2018; Arora et al., 2020). To mimic the few-shot setting, we follow previous work and subsample each dataset to a specific number of examples per class (Gao et al., 2021). When referring to a dataset, we use the suffix -k (e.g., **clinc**-k) to indicate that the dataset has been subsampled to k examples per class[2]. Following previous work, we omit out-of-scope utterances (included in **clinc**, **curekart**, **powerplay**, and **mattress**). Dataset statistics are reported in Table 1.

[2]For any class that has fewer than k examples, we select all examples of c.

Model Selection We study true few-shot text classification, i.e., few-shot learning in which no validation data is provided. Given the importance of selecting suitable hyperparameters for state-of-the-art models, we experiment with the following approaches for constructing a validation set:

- **HoldOut** - 20% of the training data (per class) is held out and used for validation. This resembles a typical workflow for non-few-shot settings.
- **Train** - use the training set as the validation set. This effectively selects the model with the lowest training loss; overfitting is expected.
- **PanGeA** - use a generator trained by PanGeA to construct 20 validations examples per class[3].
- **EDA** - similar to the previous approach but use task-agnostic data augmentation for generation (Wei and Zou, 2019).
- **Test** - use the test set as the validation set; a competitive yet unrealistic baseline included for completeness.

Procedure: We begin constructing 10 unique *variants* of each dataset (e.g., $\{$**clinc**-$5^{(1)}, \ldots,$ **clinc**-$5^{(10)}\}$). We do this by sampling a unique training set for each variant from the corresponding full dataset. None of the variants have any examples for validation; all variants use the same (original) test set. For all variants, we use each of the methods described above to construct a unique validation set. For each variant and validation set pair, we initiate 100 instances of training that vary only by hyperparameter configuration. In a given training episode (i.e., dataset variant, validation set, and hyperparameter configuration), after each epoch, we evaluate the model's loss on the validation set. The model with the lowest validation loss among all hyperparameter configurations is selected[4]. We report mean and standard deviation of test set accuracy for models selected via each method (e.g., EDA) across all variants of the same dataset. Since training sets differ per variant, we expect standard deviations to be high (Dodge et al., 2020). Thus, we report whether each method is significantly better than HoldOut using a one-sided Wilcoxon signed-rank test with significance level of $p = 0.05$ (Schuurmans, 2006; Wilcoxon, 1947). We perform the experiment with two training styles: FineTune, in which all model param-

[3]This value was chosen arbitrarily.

[4]For Test, we experimented with selecting models using validation accuracy, but found that it made hyperparameter optimization more difficult in a handful of cases.

	bank	clinc	curekart	powerplay	snips	mattress
classes	77	150	28	59	7	21
test examples	3080	4500	459	309	700	253

Table 1: **Number of Classes and Test Examples Per Dataset.**

k = 5	**bank**	**clinc**	**curekart**	**powerplay**	**snips**	**mattress**
HOLDOUT	$0.70_{0.01}$	$0.84_{0.01}$	$0.58_{0.06}$	$0.51_{0.04}$	$0.87_{0.02}$	$0.59_{0.05}$
TRAIN	$0.73_{0.02}*$	$0.86_{0.01}*$	$0.54_{0.06}$	$0.53_{0.06}$	$0.86_{0.03}$	$0.60_{0.05}$
PANGEA	$\mathbf{0.74_{0.01}}*$	$\mathbf{0.87_{0.02}}*$	$\mathbf{0.62_{0.06}}$	$0.54_{0.03}*$	$\mathbf{0.89_{0.01}}*$	$0.64_{0.05}*$
EDA	$\mathbf{0.74_{0.01}}*$	$\mathbf{0.87_{0.01}}*$	$0.58_{0.06}$	$\mathbf{0.55_{0.03}}*$	$0.88_{0.03}$	$\mathbf{0.65_{0.06}}*$
TEST	$0.76_{0.01}*$	$0.88_{0.01}$	$0.66_{0.04}*$	$0.57_{0.03}*$	$0.91_{0.01}*$	$0.69_{0.03}*$
k = 10						
HOLDOUT	$0.81_{0.01}$	$0.91_{0.00}$	$0.71_{0.04}$	$0.55_{0.02}$	$0.91_{0.02}$	$0.68_{0.03}$
TRAIN	$0.81_{0.02}$	$0.90_{0.01}$	$0.72_{0.05}$	$0.58_{0.02}*$	$0.91_{0.02}$	$0.67_{0.04}$
PANGEA	$0.83_{0.01}*$	$0.91_{0.01}*$	$\mathbf{0.73_{0.03}}*$	$\mathbf{0.60_{0.01}}*$	$\mathbf{0.92_{0.01}}$	$0.70_{0.02}$
EDA	$\mathbf{0.84_{0.01}}*$	$\mathbf{0.92_{0.01}}*$	$0.72_{0.04}$	$\mathbf{0.60_{0.02}}*$	$\mathbf{0.92_{0.02}}*$	$\mathbf{0.73_{0.02}}*$
TEST	$0.84_{0.01}*$	$0.92_{0.01}*$	$0.77_{0.03}*$	$0.57_{0.15}*$	$0.93_{0.01}*$	$0.74_{0.02}*$

Table 2: **Test Set Accuracy, FINETUNE, $\mathbf{k = \{5, 10\}}$.** Mean and standard deviation test set accuracy of models selected in the FINETUNE setting. **Bolded** text indicates the highest mean per dataset (other than TEST); asterisk (*) indicates improvement over HOLDOUT is statistically significant (1-sided Wilcoxon signed rank test, $p = 0.05$).

eters are trained, and FROZEN, in which only the last layer parameters are trained. Results for the FROZEN setting are reported in the Appendix (Section B.3). In all experiments, we use the HuggingFace `roberta-base` model optimized with the AdamW optimizer (Wolf et al., 2019; Loshchilov and Hutter, 2018).

Hyperparameters: We tune 4 hyperparameters: learning rate, weight decay, dropout among hidden units, and dropout among classifier units. We employ Optuna—a hyperparameter optimization library (Akiba et al., 2019). For each dataset variant and validations set, we allot Optuna a budget of 100 trials (i.e., unique hyperparameter configurations) with trial pruning turned on. All models are trained for up to 30 epochs. Hyperparameter ranges used during optimization are included in Appendix B.1.

3.2 Accuracy of Selected Model

Table 2 contains the mean and standard deviation for each model selection method on all 6 datasets for both $k = 5$ and $k = 10$ (i.e., 5 or 10 examples per class), when training in the FINETUNE setting. The results show that the generative methods (i.e., either PANGEA or EDA) achieve the highest mean accuracy on all datasets for both $k = 5$ and $k = 10$. While some error bars overlap, high standard deviations are anticipated since every dataset variant has a unique training set. Despite this variation, improvements of PANGEA and EDA over HOLDOUT are statistically significant in 4 or 5 datasets out of 6 for $k = 5$ and $k = 10$. Moreover, for PANGEA on **curekart**-5, our statistical test yields a value of $p = 0.0654$, only narrowly missing the $p = 0.05$ threshold. TRAIN only achieves 1 or 2 such improvements. For EDA and PANGEA in the $k = 5$ setting, improvements in mean accuracy over HOLDOUT range from 2% to 6% and as much as 8% over TRAIN. For $k = 10$, increases are more modest, but are as large as 5% over HOLDOUT and 6% over TRAIN. Note that in all FINETUNE experiments, mean accuracy of PANGEA and EDA are always greater than or equal to that of HOLDOUT. These results support the notion that validations sets constructed via PANGEA or EDA are consistent, high-performing tools for model selection in few-shot intent classification. Presentation and discussion of results for the FROZEN setting are included in Appendix B.3.

3.3 Estimating Test Set Accuracy

While selecting the best performing model among a set is a crucial step of machine learning workflows, an accurate estimate of the selected model's performance on test data is a significant factor in

k = 5	bank	clinc	curekart	powerplay	snips	mattress
HOLDOUT	$\mathbf{0.03_{0.02}}$	$\mathbf{0.04_{0.02}}$	$0.18_{0.07}$	$0.36_{0.04}$	$0.12_{0.04}$	$0.25_{0.09}$
TRAIN	$0.27_{0.02}$	$0.14_{0.01}$	$0.46_{0.06}$	$0.47_{0.06}$	$0.14_{0.03}$	$0.40_{0.05}$
PANGEA	$0.18_{0.02}$	$0.20_{0.02}$	$\mathbf{0.14_{0.07}}$	$\mathbf{0.19_{0.03}}$	$\mathbf{0.03_{0.02}}$	$\mathbf{0.07_{0.05}}$
EDA	$0.24_{0.01}$	$0.09_{0.01}$	$0.39_{0.06}$	$0.40_{0.03}$	$0.12_{0.03}$	$0.30_{0.06}$
k = 10						
HOLDOUT	$\mathbf{0.03_{0.01}}$	$\mathbf{0.03_{0.01}}$	$0.16_{0.06}$	$0.34_{0.02}$	$0.09_{0.02}$	$0.23_{0.05}$
TRAIN	$0.19_{0.02}$	$0.10_{0.01}$	$0.28_{0.05}$	$0.42_{0.02}$	$0.09_{0.02}$	$0.33_{0.04}$
PANGEA	$0.34_{0.01}$	$0.28_{0.01}$	$\mathbf{0.05_{0.04}}$	$\mathbf{0.11_{0.02}}$	$0.09_{0.04}$	$\mathbf{0.04_{0.03}}$
EDA	$0.14_{0.01}$	$\mathbf{0.03_{0.01}}$	$0.25_{0.04}$	$0.36_{0.02}$	$\mathbf{0.08_{0.02}}$	$0.21_{0.02}$

Table 3: **Model Fidelity, FINETUNE, k = $\{5, 10\}$.** The mean and standard deviation of the absolute difference between validation and test set accuracy of the selected model. **Bolded** text indicates the lowest mean per dataset.

determining whether the model is eligible for deployment. That is, if the best performing model in a set performs poorly, that model is unfit for deployment. We underscore that test accuracy is not necessarily indicative of a model's ability to generalize, and that other evaluations, e.g., of the model's likelihood to cause harm, must also be carried out before determining if that model is appropriate for use (Ribeiro et al., 2020).

3.3.1 Validation Accuracy of Selected Models

To this end, we measure the extent to which validation accuracy is a faithful estimator of test set accuracy for the methods discussed above. In Table 3 we report the mean and standard deviation of the absolute difference between validation and test set accuracy for models selected via each method in the FINETUNE setting for $k = 5$ and $k = 10$. If the magnitude of the difference of a model's validation and test set accuracy is small, we say that the model provides a *high fidelity* estimate of test accuracy. The Table shows that PANGEA leads to the highest fidelity estimates for 4 of 6 data sets with $k = 5$ and 3 out of 6 data sets for $k = 10$. In many of these cases, PANGEA improves over the next best approach by more than 3x.

While the other methods yield higher fidelity estimates of test accuracy for **clinc** and **bank-**10, we note that the fidelity of these methods is highly correlated with test set accuracy. Figure 2 plots test accuracy vs. the mean absolute difference between validation and test accuracy for all methods and datasets, for the $k = 5$ variants and FINETUNE setting. Unsurprisingly, for TRAIN, the difference between validation and test accuracy is perfectly anti-correlated with test accuracy, i.e., when test accuracy is high so is validation accuracy, but valida-

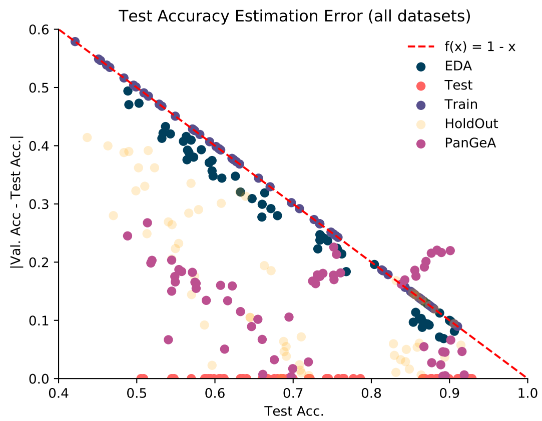


Figure 2: **Test Accuracy vs. Test Accuracy Estimation Error, FINETUNE, $k = 5$.** Test set accuracy vs. the mean absolute difference of validation and test accuracy (i.e., error).

tion accuracy remains high even when test accuracy is low. This is unsurprising since the trained models consistently fit the training data, and thus validation accuracy on TRAIN is always near 100%. This makes TRAIN unreliable with respect to fidelity—since fidelity is entirely dependent on test set accuracy, which is unknown.

Both EDA and HOLDOUT exhibit similar trends. For EDA, generations closely resemble the training data since the generations are constructed via simple perturbations. Thus, a model that perfectly fits the training data is likely to fit the EDA examples. For HOLDOUT, the difference between validation and test accuracy is also somewhat anticorrelated with test accuracy. Again, this is because the validation data is sampled directly from the training data. For $k = 5$, Because the validation set is small, fidelity has higher variance, which

	FINETUNE-5	FINETUNE-10
HOLDOUT	0.13324	0.11809
TRAIN	0.30141	0.23781
PANGEA	**0.00367**	**0.08993**
EDA	0.24572	0.18091

Table 4: **RMSE of Validation Accuracy, FINETUNE.** The root mean square error with respect to validation accuracy and test set accuracy for all methods and training regimes. RMSE is computed from all hyperparameter configurations, all epochs, and all datasets. **Bolded** text indicates the lowest RMSE per condition. Note that RMSE for TEST is 0.

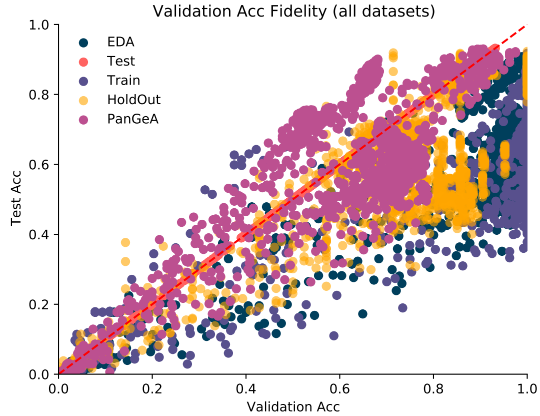


Figure 3: **Fidelity of Validation Accuracy, FINETUNE, k = 5.** Validation set accuracy versus test set accuracy for all hyperparamter configurations, all epochs, and for all datasets.

leads to the dampened anti-correlation. We note that the anti-correlation is more pronounced for $k = 10$ because the corresponding validation sets are twice as large and thus yield fidelity with lower variance (the corresponding visualization appears in Figure 5, located in Appendix B.3). We conclude that accuracy on validation sets constructed by PANGEA are the most reliable approximations of test set accuracy among all methods tested. However, even for PANGEA, the difference between validation and test accuracy is often too high (in many cases greater than 10%) to make for a useful estimate that can be leveraged in deployment decisions.

3.3.2 All Hyperparameter Configurations

We examine the difference between validation and test accuracy for all hyperparameter configurations, all datasets, and all epochs—rather than just for the selected models. This gives a sense of how accurate test set accuracy can be predicted by validation accuracy, regardless of how hyperparameters are chosen and how a model is selected. We report the root mean square error (RMSE) between validation accuracy and test set accuracy in Table 4. The Table shows that PANGEA yields the highest fidelity estimates of test set accuracy (i.e., lowest RMSE) for both $k = 5$ and $k = 10$.

For a more detailed view, we visualize the correlation between validation accuracy and test set accuracy in Figure 3. Note that, while Figure 2 visualizes performance of selected models only, Figure 3 visualizes performance for all models (i.e., all hyperparamter configurations and training epochs). The Figure shows that for all hyperparameter configurations and training epochs, accuracy on validation sets constructed by PANGEA roughly matches test set accuracy. On the other hand, the other methods only match test set accuracy when validation accuracy is low, but consistently overestimate test set accuracy as validation accuracy increases. We include a similar figure for $k = 10$ in Figure 4, located in Appendix B.2.

4 Related Work

Our work is a first systematic study of validation set construction to support model selection in true few-shot intent classification. Prior to our work, two other pieces have leveraged generative models to construct validation data. In Datasets from Instructions (DINO), a pre-trained GPT2-XL is prompted to generate labeled sentence pairs to support learning improved sentence embeddings (Schick and Schütze, 2021). The set of generated pairs is split into training and validation sets. In this work, the validation set is used to determine when to (early) stop training, but it is unclear whether it is also used to select among a range of hyperparameter configurations. In our study, we use constructed validation sets to select 4 important hyperparameters, in addition to early stopping. The tasks we focus on are domain-specific, whereas DINO is aimed and learning better general-purpose sentence embeddings–where it may be easier to generate relevant data for validation.

The second piece studies prompt order for "in-context learning" (Brown et al., 2020), i.e., when the model is given a handful of examples of a task at inference time but no weights are updated. The authors find that the order of the examples in the prompt used for in-context learning can significantly affect results (fluctuations between state-of-the-art and random chance performance were observed) (Lu et al., 2022). To alleviate this high sensitivity in true few-shot settings, the authors generate an unlabeled validation set with a large pre-trained language model and use the set to select prompt orders via a proposed entropy-based method. Unlike their study, we focus on the FINETUNE and FROZEN cases rather than in-context learning, because they are more practical in terms of hardware costs and thus more prevalent (Gao et al., 2021). Moreover, we select specific values of continuous hyperparamters rather than the best among small set of prompt-permutations. Finally, we point out that the proposed approach for prompt-order selection cannot be directly used to estimate test set accuracy (as we study in Section 3.3).

A central component of our work is our proposed PANGEA algorithm. Like our approach, previous work makes use of a sequence-to-sequence model for generation, but unlike ours, that work focuses on filling in delexicalized utterances (Hou et al., 2018). Our use of an utterance to prompt the generator is similar in spirit to work on Example Extrapolation (EX2) (Lee et al., 2021). Whereas their work focuses on uneven amounts of data per class, we focus on true few-shot learning. Unlike EX2, we only provide the generator with a single utterance, rather than many. Using a single utterance to prompt the generator is also similar to work on using demonstrations (Gao et al., 2021), but in that work, training examples are concatenated to the input during training and inference. We also provide the PANGEA-trained generator with guide words, which is inspired by previous work on decoding (Pascual et al., 2021).

While we experiment with a handful of approaches, there is a large and growing literature on data augmentation for NLP. We briefly touch on some recently proposed methods, but refer interested readers to a survey on the subject (Feng et al., 2021). Most data augmentation algorithms can be roughly categorized as either retrieval (Du et al., 2021), perturbation (Wei and Zou, 2019), feature (Kumar et al., 2019; Sun et al., 2020; Wei, 2021), or generation-based (Wang et al., 2021; Kumar et al., 2020; He et al., 2021; Yang et al., 2020). Some work focuses on counterfactual augmentation (Kaushik et al., 2020; Joshi and He, 2022); likewise, generating minimally perturbed training examples with different labels (Zhou et al., 2022). In the literature, augmentation is generally employed as a tool for improving test set accuracy. But a recent studies explore augmentation for mitigating gender stereotypes (Zhao et al., 2018; Zmigrod et al., 2019; Maudslay et al., 2019; Webster et al., 2020). Unlike our work, virtually all previous studies focused on training set augmentation rather than validation set construction.

5 Conclusion

In this work we study true-few shot classification, i.e., few-shot classification where no validation set is provided for model selection. We experiment with constructing validation sets via data augmentation, and by leveraging the provided few-shot data. Our results reveal that the synthetic validation sets—constructed by EDA or our proposed method, PANGEA—consistently yield selected models with the higher test accuracy than validation sets comprised of the few-shot data. Moreover, PANGEA is the only method for which validation accuracy provides a reliable, high fidelity estimate of test set accuracy.

6 Limitations

In this work, we study various methods of validation set construction for the true few-shot setting. While we show that methods of data augmentation can be successfully utilized, our experiments only deal with few-shot intent classification. All of our experiments are conducted on English language data sets. Additionally, our experiments include subsampled data sets with either 5 or 10 examples per class (when enough examples per class exists), but we do not experiment with (intentionally) unbalanced data sets. Moreover, we only experiment with the RoBERTa model. We choose RoBERTa because it is high-performing and ubiquitous (and therefore admits comparison to other work), but we acknowledge that better models exist and may provide different results. Despite these limitations, we believe that our results are sound and likely to generalize to models aside from RoBERTa. Finally, we do not experiment with in-context learning methods (i.e., prompting with GPT-3); but we argue that the FROZEN and FINETUNE settings are prominent training paradigms that are currently accessible to many more people.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *International Conference on Knowledge Discovery and Data Mining*.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! *Association for the Advancement of Artificial Intelligence*.

Gaurav Arora, Chirag Jain, Manas Chaturvedi, and Krupal Modi. 2020. Hint3: Raising the bar for intent detection in the wild. In *Workshop on Insights from Negative Results in NLP*.

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. Flex: Unifying evaluation for few-shot nlp. *Advances in Neural Information Processing Systems*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Workshop on NLP for ConvAI*. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv:1805.10190*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv:2002.06305*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. 2021. Self-training improves pre-training for natural language understanding. In *Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing*. Association for Computational Linguistics.

Xuanli He, Islam Nassar, Jamie Kiros, Gholamreza Haffari, and Mohammad Norouzi. 2021. Generate, annotate, and learn: Nlp with synthetic text. *arXiv:2106.06168*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-Sequence data augmentation for dialogue language understanding. In *International Conference on Computational Linguistics*.

Nitish Joshi and He He. 2022. An investigation of the (in)effectiveness of counterfactually augmented data. In *Association for Computational Linguistics*. Association for Computational Linguistics.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with Counterfactually-Augmented data. In *International Conference on Learning Representations*.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2021. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Workshop on Life-long Learning for Spoken Language Systems*.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019. A closer look at feature space data augmentation for Few-Shot intent classification. In *Workshop on Deep Learning Approaches for Low-Resource NLP*.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation. *arXiv:2102.01335*.

Shayne Longpre, Yu Wang, and Chris DuBois. 2020. How effective is Task-Agnostic data augmentation for pretrained transformers? In *Findings of the Association for Computational Linguistics: EMNLP*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming Few-Shot prompt order sensitivity. In *Association for Computational Linguistics*.

Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell, and Simone Teufel. 2019. It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. In *Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in Neural Information Processing Systems*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Association for Computational Linguistics*.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Empirical Methods in Natural Language Processing*.

Dale Schuurmans. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-Transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv:2109.09193*.

Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, Ed Chi, and Slav Petrov. 2020. Measuring and reducing gendered correlations in pre-trained models. *arXiv:2010.06032*.

Jason Wei. 2021. Good-Enough example extrapolation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

Frank Wilcoxon. 1947. Probability tables for individual comparisons by ranking methods. *Biometrics*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Association for Computational Linguistics: Human Language Technologies*, pages 15–20.

Jing Zhou, Yanan Zheng, Jie Tang, Li Jian, and Zhilin Yang. 2022. FlipDA: Effective and robust data augmentation for few-shot learning. In *Association for Computational Linguistics*. Association for Computational Linguistics.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Association for Computational Linguistics*.

Appendix

A Question Extraction

We extract question from Common Crawl—a large-scale archive of crawled webpages. We use a combination of 10 Common Crawl dumps from 2020 and 2021, which includes 33 billion webpages. To detect question and answer (QA) content nested in raw webpages, we leverage structured markup for QA[5] and FAQ[6] pages. This markup is widely used, and facilitates the display of QA result previews along with search results (e.g., google search).

Naive search in billions of webpages in costly. Therefore, we first perform a fast regex-based search that yields approximately 26 million matching HTML pages. After parsing the resulting pages, we are able to extract approximately 71 million QA and FAQ data snippets. We then post-process the results by removing badly formatted snippets where questions or answers cannot be automatically recovered, pruning empty question or answer bodies, and performing language detection to identify English QA pairs. The result is 27.7 million English pairs. We group the English questions by page and randomly select 200k question pairs for training such that both questions appeared on the same page.

B Experiments

All experiments on run on 2 NVIDIA Ampere (A100) GPUs.

B.1 Hyperparameter Ranges

For hyperparameter optimization, we use Optuna (Akiba et al., 2019). Optuna allows a practitioner to identify the hyperparameters over which to conduct the search, as well as the allowable ranges. In our experiments, Optuna tunes the following 4 parameters with the following ranges:

1. learning rate, $[0.00001, 0.1]$;
2. weight decay, $[0.0, 0.1]$;
3. dropout among hidden units, i.e., `hidden_dropout_prob`, $[0.0, 0.5]$; and
4. dropout among classification head units, i.e., `classifier_dropout`, $[0.0, 1.0]$.

[5] https://developers.google.com/search/docs/advanced/structured-data/qapage

[6] https://developers.google.com/search/docs/advanced/structured-data/faqpage

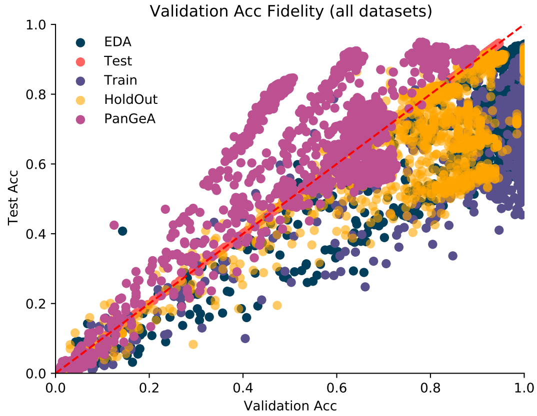


Figure 4: **Fidelity of Validation Accuracy, FINETUNE, k = 10.** Validation set accuracy versus test set accuracy for all hyperparameter configurations, all epochs, and for all datasets.

Optuna performs 100 trials (each trial may be pruned if the corresponding hyperparameters are deemed unlikely to yield a high performing model. New configurations are sampled using the `TPESampler` (the random seed is set to 37). Training in a full trial lasts for 30 epochs.

B.2 Model Fidelity in the FINETUNE Setting

Figure 4 visualizes validation accuracy vs. test accuracy for all methods in the FINETUNE setting with $k = 10$. Like in the case of $k = 5$, accuracy on validation sets constructed by PANGEA appear to be better correlated with test accuracy than either TRAIN or EDA, which both consistently overestimate test set accuracy. In the $k = 10$ case, it appears that PANGEA tends to more strongly underestimate test set accuracy.

Figure 5 plots test accuracy vs. the mean absolute difference between validation and test accuracy for all methods and datasets, for the $k = 10$ variants and FINETUNE setting. As in the case of $k = 5$, for TRAIN, the difference between validation and test accuracy is perfectly anti-correlated with test accuracy. EDA and HOLDOUT are also strongly anti-correlated with test set accuracy. This makes these three methods unreliable with respect to fidelity—since fidelity is entirely dependent on test set accuracy, which is unknown. On the other hand, PANGEA is not anti-correlated with test accuracy, but exhibits some low fidelity estimates.

k = 5	bank	clinc	curekart	powerplay	snips	mattress
HOLDOUT	$0.34_{0.01}$	$0.52_{0.01}$	$0.28_{0.05}$	$0.30_{0.03}$	$0.79_{0.07}$	$0.32_{0.03}$
TRAIN	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$\mathbf{0.37_{0.04}}*$	$\mathbf{0.33_{0.03}}*$	$0.82_{0.06}$	$\mathbf{0.38_{0.03}}*$
PANGEA	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$0.34_{0.04}*$	$0.32_{0.02}*$	$\mathbf{0.84_{0.02}}*$	$\mathbf{0.38_{0.03}}*$
EDA	$\mathbf{0.39_{0.01}}*$	$\mathbf{0.60_{0.01}}*$	$0.35_{0.04}*$	$0.32_{0.02}$	$0.80_{0.09}$	$\mathbf{0.38_{0.03}}*$
TEST	$0.39_{0.01}*$	$0.60_{0.01}*$	$0.32_{0.04}*$	$0.32_{0.02}$	$0.84_{0.02}*$	$0.38_{0.03}*$
k = 10						
HOLDOUT	$0.54_{0.01}$	$0.75_{0.01}$	$0.48_{0.04}$	$0.33_{0.02}$	$0.84_{0.04}$	$0.38_{0.02}$
TRAIN	$\mathbf{0.68_{0.01}}*$	$\mathbf{0.82_{0.01}}*$	$\mathbf{0.54_{0.04}}*$	$\mathbf{0.36_{0.02}}*$	$\mathbf{0.88_{0.01}}*$	$0.44_{0.02}*$
PANGEA	$0.59_{0.02}*$	$0.77_{0.01}*$	$0.53_{0.05}*$	$\mathbf{0.36_{0.02}}*$	$\mathbf{0.88_{0.01}}*$	$0.44_{0.02}*$
EDA	$0.61_{0.03}*$	$0.81_{0.01}*$	$0.55_{0.03}*$	$\mathbf{0.36_{0.02}}*$	$\mathbf{0.88_{0.01}}*$	$\mathbf{0.46_{0.02}}*$
TEST	$0.63_{0.03}*$	$0.79_{0.02}*$	$0.56_{0.04}*$	$0.38_{0.03}*$	$0.89_{0.01}*$	$0.47_{0.02}*$

Table 5: **Test Set Accuracy, FROZEN, $k = \{5, 10\}$.** Mean and standard deviation test set accuracy of models selected in the FROZEN setting. **Bolded** text indicates the highest mean per dataset (other than TEST); asterisk (*) indicates improvement over HOLDOUT is statistically significant (1-sided Wilcoxon signed rank test, $p = 0.05$).

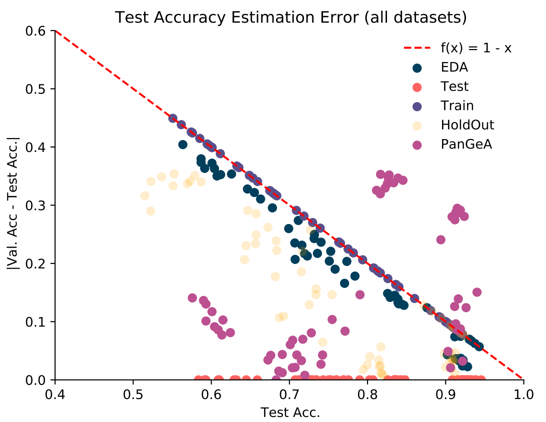


Figure 5: **Test Accuracy vs. Test Accuracy Estimation Error, FINETUNE, $k = 10$.** Test set accuracy vs. the mean absolute difference of validation and test accuracy (i.e., error).

B.3 Model Selection in the FROZEN Setting

In this section we present the results of model selection when training is carried out in the FROZEN setting. The FROZEN case (also known as the "linear probing" setting) is common when latency and/or computing cost are constrained. Moreover, FROZEN training has been shown to generalize better to out-of-distribution data than FINETUNE training when pre-trained representations are "good" (Kumar et al., 2021). This is relevant to the few-shot domain where most data may be considered out-of-distribution because of the scarcity of training data.

The results in the FROZEN setting are somewhat different than the FINETUNE setting. We begin with Table 5, which contains the test set accuracy of selected models. First, we note that accuracy is universally lower than in the FINETUNE setting. This is because many fewer parameters are being trained. Additionally, there are many more statistically significant improvements over HOLDOUT. This indicates that holding out training data for validation is particularly costly in the FROZEN setting. Among the methods, we point out that PANGEA is the only method to exhibit statistically significant improvements for every dataset for both $k = 5$ and $k = 10$.

The Table 5 reveals two surprising phenomena. First, TRAIN is very competitive; with many statistically significant improvements over HOLDOUT and often achieving the highest mean accuracy among all methods. Second, TEST does not achieve the highest accuracy for a handful of datasets. Upon inspection, we find that hyperparameter optimization is the cause of both phenomena. Specifically, some validation sets lead to more effective hyperparameter optimization. As an example, consider Figure 9a. Each circle in the Figure corresponds to the test set accuracy of a selected model for a specific dataset variant. Recall that a selected model is defined by a set of hyperparameters and an epoch (identified by hyperparameter optimization to achieve the lowest validation loss). Each star (*) in the Figure is the maximum achievable test set accuracy for a model trained with the same hyperparameters. Therefore, for a given set of hyperparameters, if the epoch in which the smallest validation loss is achieved is the same

	FROZEN-5	FROZEN-10
HOLDOUT	0.05476	**0.04637**
TRAIN	0.34020	0.26431
PANGEA	**0.00273**	0.09728
EDA	0.23756	0.16804

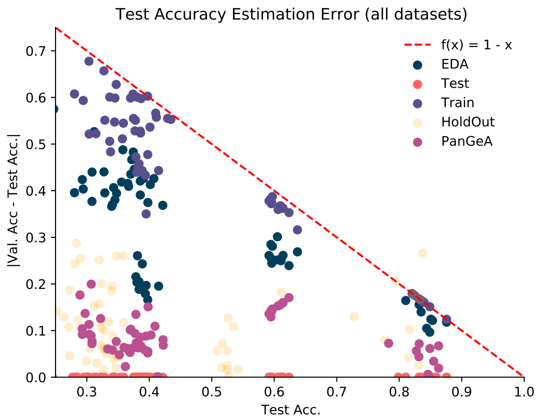
Table 6: **RMSE of Validation Accuracy, FROZEN.** The root mean square error with respect to validation accuracy and test set accuracy for all methods and training regimes. RMSE is computed from all hyperparameter configurations, all epochs, and all datasets. **Bolded** text indicates the lowest RMSE per condition. Note that RMSE for TEST is 0.

as the epoch where the highest test set accuracy is achieved, then the circle and star corresponding to that dataset variant will have the same y-value. Mean test set accuracy for the selected model and mean of the maximum possible test set accuracy are also visualized. The Figure reveals that, for **bank-**10 in the FROZEN setting, the best hyperparameters are found when minimizing training loss. We provide similar plots for all experimental settings and datasets for completeness.

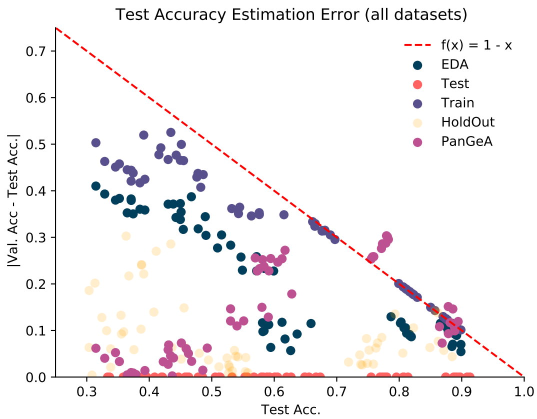
B.4 Fidelity in the FROZEN Setting

In Table 7 we report the mean and standard deviation of the absolute difference between validation and test set accuracy for models selected via each method in the FROZEN setting for $k = 5$ and $k = 10$. The Table shows that PANGEA leads to the highest fidelity estimates for 4 of 6 data sets with $k = 5$; for the remaining two datasets, it achieves the second highest fidelity. When $k = 10$, PANGEA is best in 2 out of 6 data sets. Unlike the FINETUNE case, HOLDOUT is more competitive when training in the FROZEN setting. Both TRAIN and EDA provide unreliable (and low fidelity) estimates since their fidelity is highly correlated with test set accuracy (as in the FINETUNE case). The correlation is visualized in Figure 6.

As in the FINETUNE case, we examine the difference between validation and test accuracy for all hyperparameter configurations, all datasets, and all epochs. We report the root mean square error (RMSE) between validation accuracy and test set accuracy in Table 6. The Table shows that PANGEA yields the highest fidelity estimates of test set accuracy (i.e., lowest RMSE) for $k = 5$ and second highest when $k = 10$. Conversely, HOLDOUT yields the second highest fidelity estimates for $k = 5$ and the highest fidelity estimates when



(a) $k = 5$



(b) $k = 10$

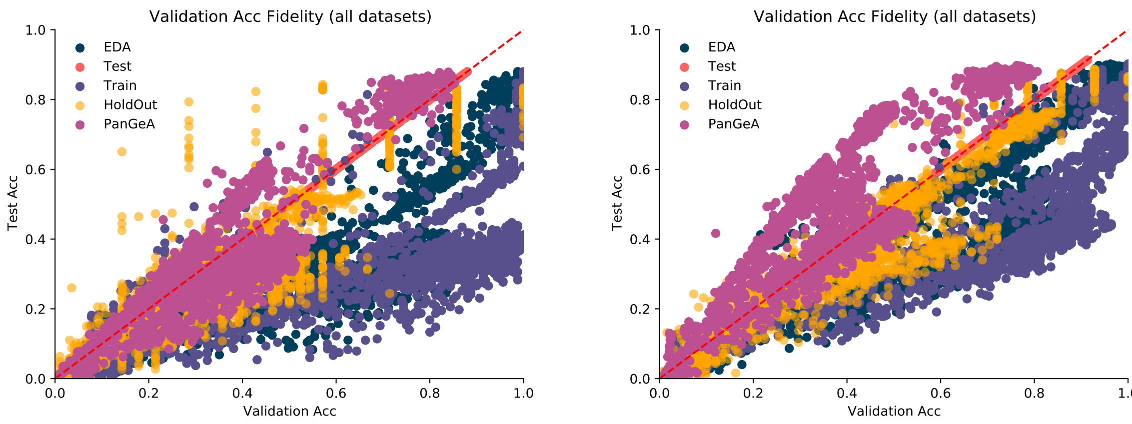
Figure 6: **Test Accuracy vs. Absolute Difference of Validation and Test Accuracy, FROZEN, $k = \{5, 10\}$.**

$k = 10$.

Finally, in Figure 7 we visualize validation accuracy vs. test accuracy for selected models in the FROZEN setting for all hyperparameter configurations and training epochs. As in the FINETUNE setting, we find that accuracy on validation sets constructed by PANGEA and HOLDOUT are most highly correlated with test set accuracy. For $k = 5$, HOLDOUT exhibits high variance. Again, both EDA and TRAIN overestimate test accuracy.

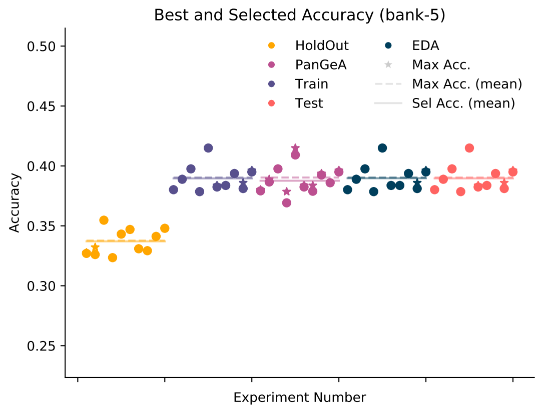
$\mathbf{k = 5}$	**bank**	**clinc**	**curekart**	**powerplay**	**snips**	**mattress**
HOLDOUT	$\mathbf{0.05_{0.02}}$	$\mathbf{0.05_{0.04}}$	$0.18_{0.06}$	$0.15_{0.07}$	$0.14_{0.07}$	$0.13_{0.08}$
TRAIN	$0.44_{0.03}$	$0.37_{0.01}$	$0.60_{0.04}$	$0.54_{0.04}$	$0.18_{0.05}$	$0.57_{0.04}$
PANGEA	$0.07_{0.02}$	$0.15_{0.01}$	$\mathbf{0.09_{0.05}}$	$\mathbf{0.07_{0.02}}$	$\mathbf{0.04_{0.03}}$	$\mathbf{0.11_{0.04}}$
EDA	$0.21_{0.03}$	$0.26_{0.01}$	$0.47_{0.05}$	$0.40_{0.02}$	$0.16_{0.07}$	$0.41_{0.02}$
$\mathbf{k = 10}$						
HOLDOUT	$\mathbf{0.02_{0.01}}$	$\mathbf{0.05_{0.02}}$	$\mathbf{0.07_{0.04}}$	$0.10_{0.06}$	$\mathbf{0.09_{0.04}}$	$0.21_{0.08}$
TRAIN	$0.32_{0.01}$	$0.18_{0.01}$	$0.37_{0.03}$	$0.45_{0.02}$	$0.12_{0.01}$	$0.48_{0.03}$
PANGEA	$0.25_{0.01}$	$0.28_{0.02}$	$0.11_{0.05}$	$\mathbf{0.02_{0.02}}$	$0.11_{0.02}$	$\mathbf{0.05_{0.02}}$
EDA	$0.10_{0.02}$	$0.10_{0.01}$	$0.26_{0.03}$	$0.38_{0.02}$	$\mathbf{0.09_{0.02}}$	$0.35_{0.02}$

Table 7: **Model Fidelity, FROZEN, $k = \{5, 10\}$.** The mean and standard deviation of the absolute difference between validation accuracy and test set accuracy of the selected model. **Bolded** text indicates the lowest mean per dataset.

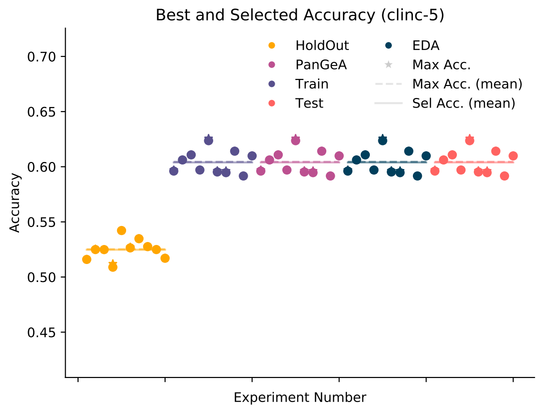


(a) $k = 5$ (b) $k = 10$

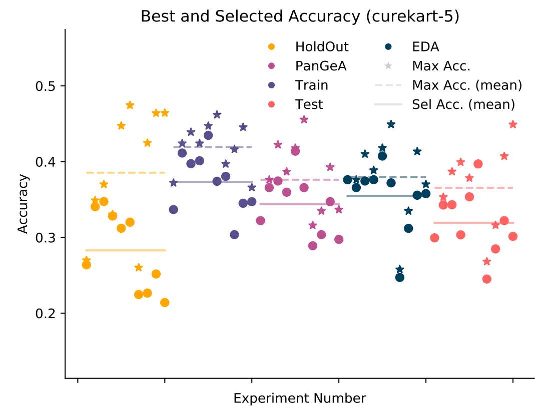
Figure 7: **Fidelity of Validation Accuracy, FROZEN, $\mathbf{k = \{5, 10\}}$.** Validation set accuracy versus test set accuracy for all hyperparameter configurations, all epochs, and for all datasets.



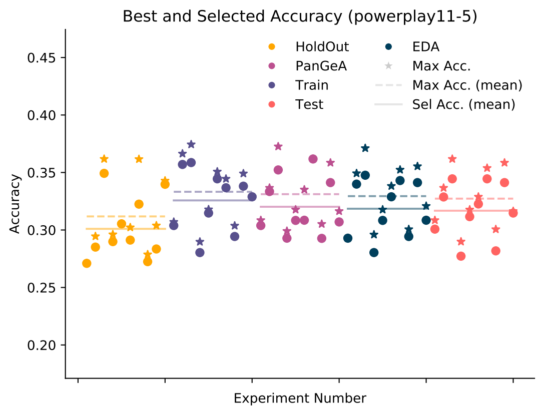
(a) **bank**-5



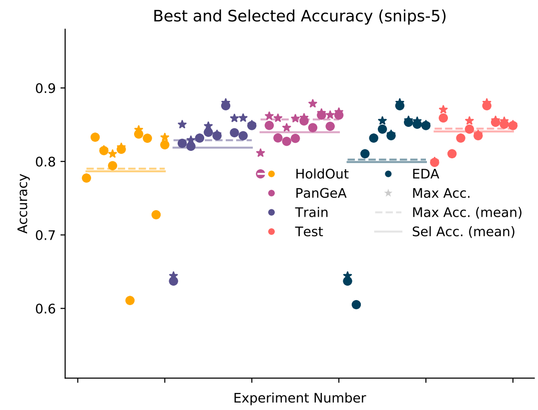
(b) **clinc**-5



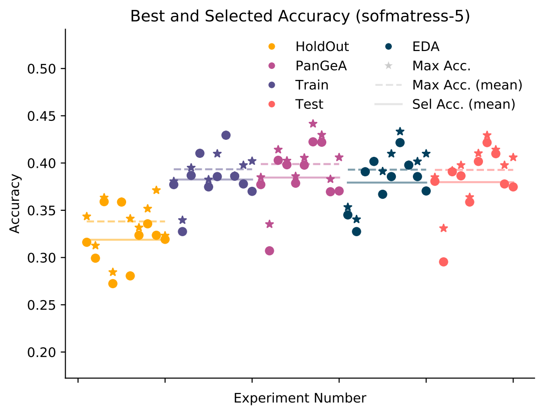
(c) **curekart**-5



(d) **powerplay**-5

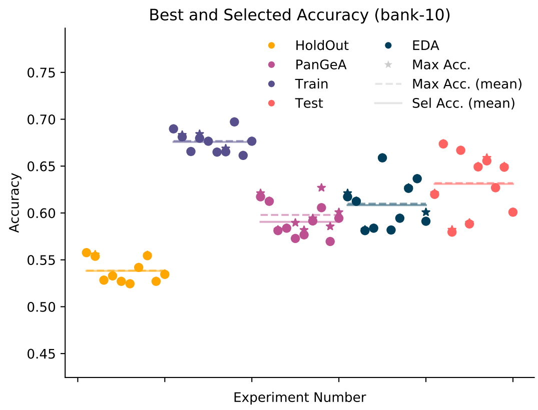


(e) **snips**-5

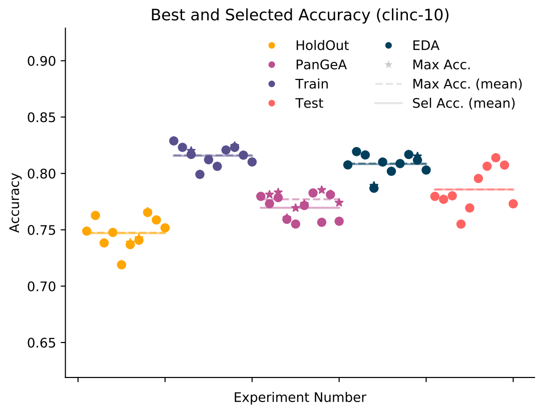


(f) **mattress**-5

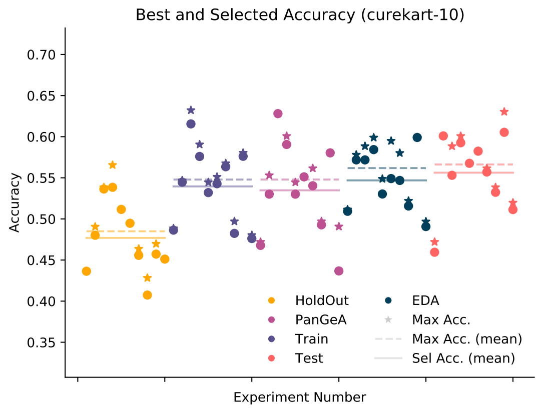
Figure 8: **Maximum and Selected Accuracy, FROZEN, $k = 5$.** Each circle represents the test set accuracy of a selected model for a single dataset variant. Stars (*) indicate the maximum accuracy achievable using the same hyperparameters. Solid lines indicate mean test accuracy of selected models; dotted lines, mean maximum accuracy.



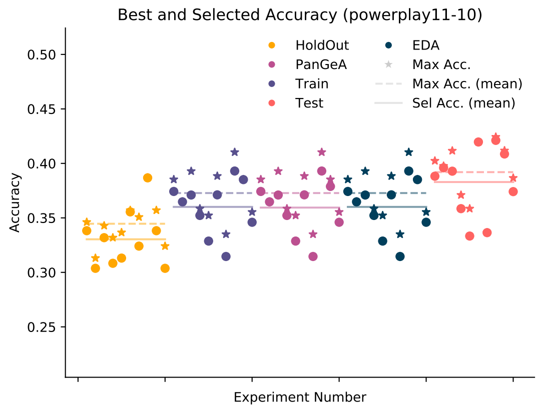
(a) **bank-**10



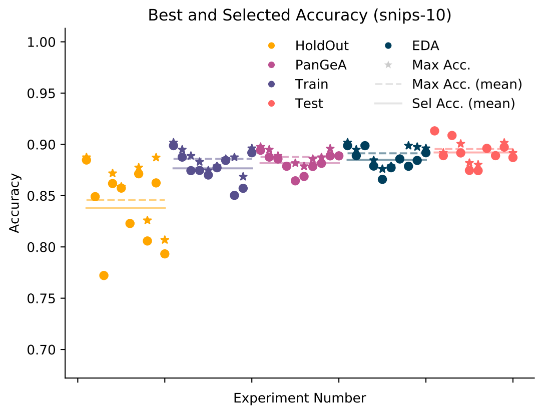
(b) **clinc-**10



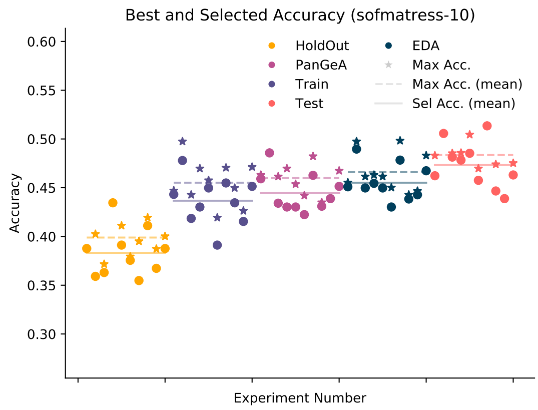
(c) **curekart-**10



(d) **powerplay-**10

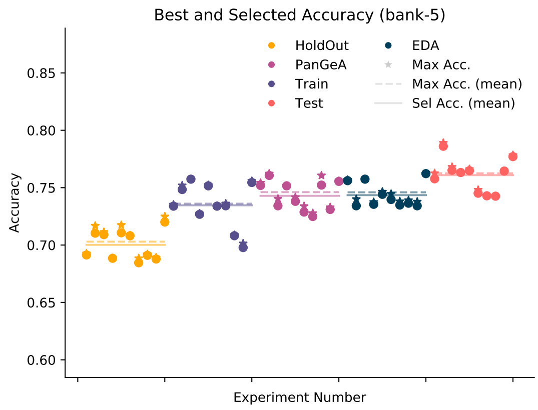


(e) **snips-**10

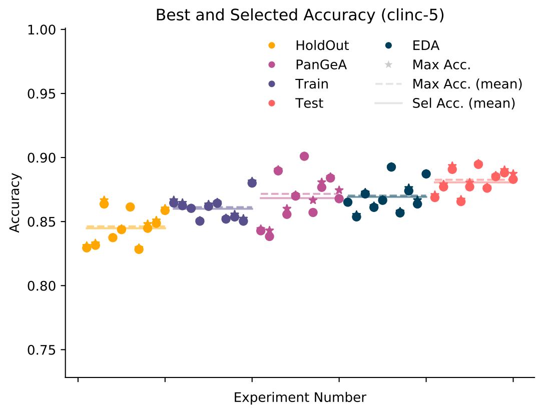


(f) **mattress-**10

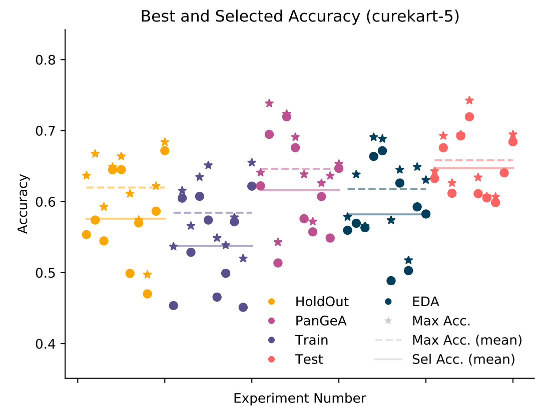
Figure 9: **Maximum and Selected Accuracy, FROZEN, $k = 10$.** Each circle represents the test set accuracy of a selected model for a single dataset variant. Stars (*) indicate the maximum accuracy achievable using the same hyperparameters. Solid lines indicate mean test accuracy of selected models; dotted lines, mean maximum accuracy.



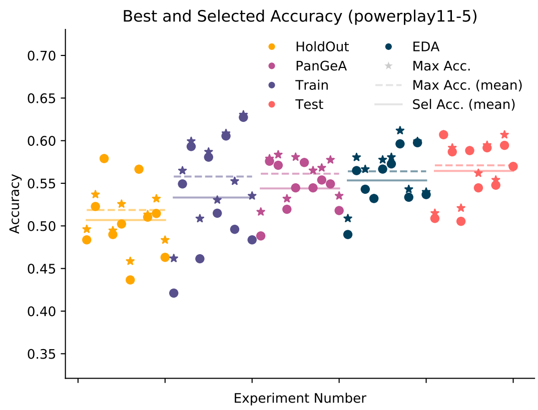
(a) **bank**-5



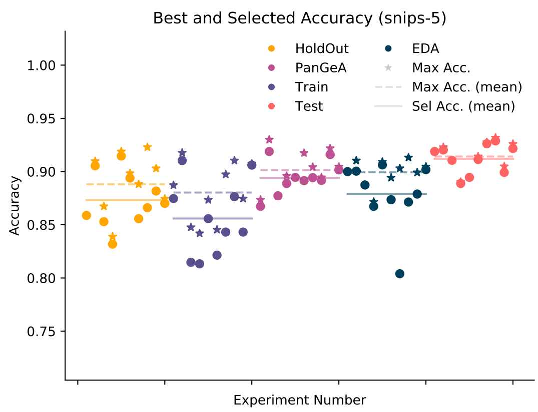
(b) **clinc**-5



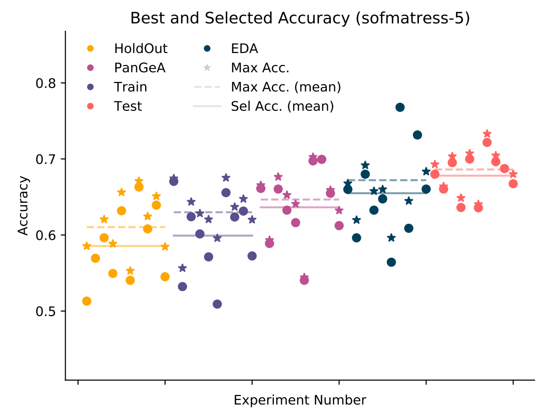
(c) **curekart**-5



(d) **powerplay**-5

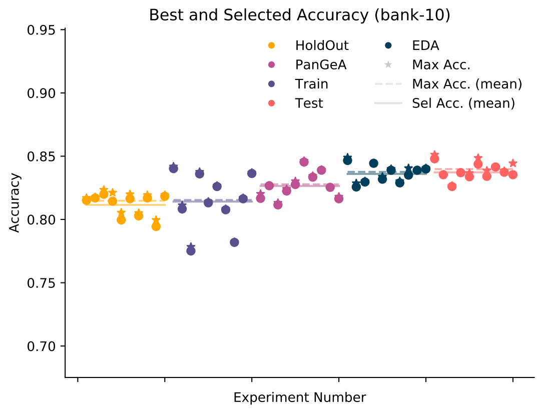


(e) **snips**-5

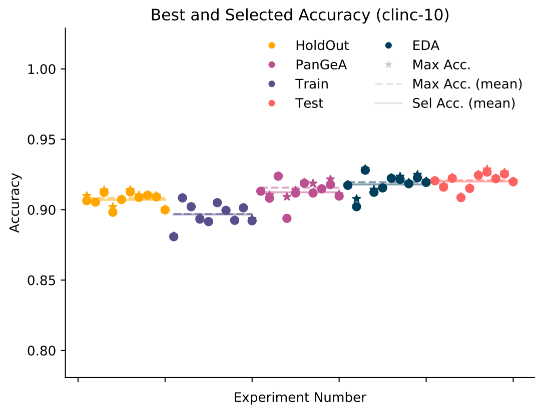


(f) **mattress**-5

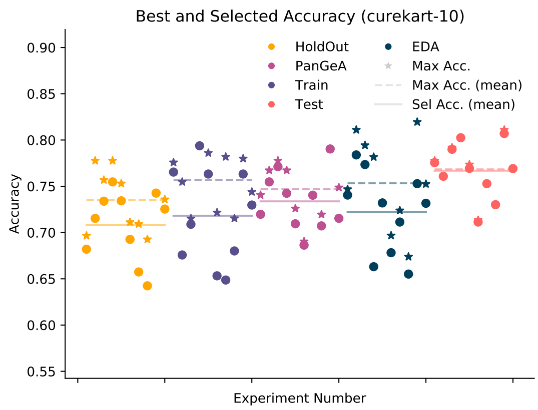
Figure 10: **Maximum and Selected Accuracy, FINETUNE, $k = 5$.** Each circle represents the test set accuracy of a selected model for a single dataset variant. Stars (*) indicate the maximum accuracy achievable using the same hyperparameters. Solid lines indicate mean test accuracy of selected models; dotted lines, mean maximum accuracy.



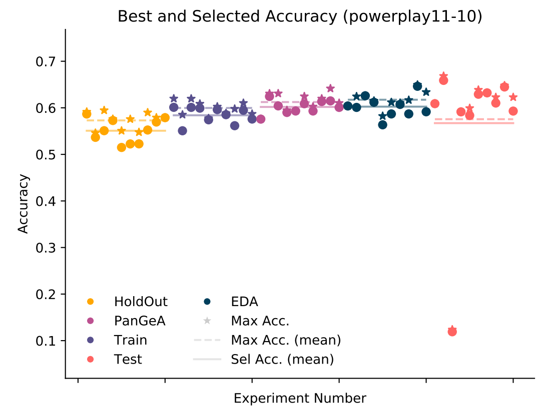
(a) **bank-**10



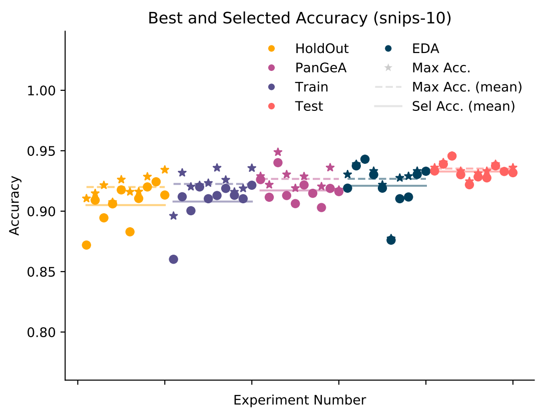
(b) **clinc-**10



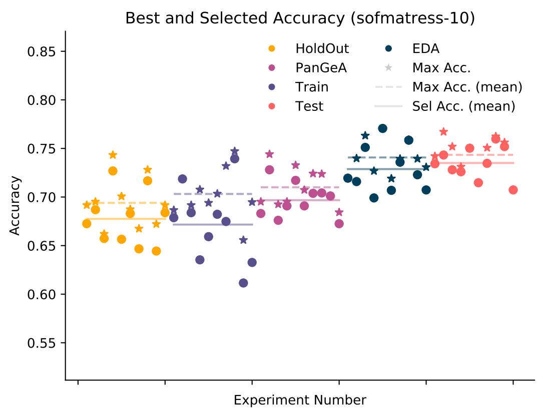
(c) **curekart-**10



(d) **powerplay-**10



(e) **snips-**10



(f) **mattress-**10

Figure 11: **Maximum and Selected Accuracy, FINETUNE, $k = 10$.** Each circle represents the test set accuracy of a selected model for a single dataset variant. Stars (*) indicate the maximum accuracy achievable using the same hyperparameters. Solid lines indicate mean test accuracy of selected models; dotted lines, mean maximum accuracy.